On the design of a family of CI pseudo-random number generators

Jacques Bahi, Xiole Fang, Christophe Guyeux, Qianxue Wang

On the design of a family of CI pseudo-random number generators

Jacques M. Bahi, Xiaole Fang, Christophe Guyeux, and Qianxue Wang
University of Franche-Comté
Computer Science Laboratory LIFC, Besançon, France
Email:jacques.bahi, xiaole.fang, christophe.guyeux, qianxue.wang@univ-fcomte.fr

***Abstract*—Chaos and its applications in the field of secure communications have attracted a lot of attention. Chaos-based pseudo-random number generators are critical to guarantee security over open networks as the Internet. We have previously demonstrated that it is possible to define such generators with good statistical properties by using a tool called "chaotic iterations", which depends on an iteration function. An approach to find update functions such that the associated generator presents a random-like and chaotic behavior is proposed in this research work. To do so, we use the vectorial Boolean negation as a prototype and explain how to modify this iteration function without deflating the good properties of the associated generator. Simulation results and basic security analysis are then presented to evaluate the randomness of this new family of generators.**

***Index Terms*—Chaos; Pseudo-random number generator; Statistical tests; Internet security; Iteration function.**

I. Introduction

Security has become a topic of increasing importance in communications because the Internet and personal communications systems are now accessible worldwide. To guarantee this security, chaotic systems have many advantages as unpredictability or disorder-like, and they are especially used when complex sequences are required [5], [8], [11]. This is why chaotic systems are frequently used to design new pseudo-random number generators (PRNGs) [5], [9].Following this approach, we have previously proposed a PRNG based on chaotic iterations. A short overview of our recent researches in this field is given hereafter.

In Ref. [3], it is proven that chaotic iterations (CIs), a suitable tool for fast computing iterative algorithms, satisfies the topological chaos property, as defined by Devaney [6]. The chaotic behavior of CIs is used in [1], to obtain an unpredictable PRNG that depends on two logistic maps. The resulted PRNG shows better statistical properties than each individual component alone. Additionally, various chaos properties have been established. These chaos properties, inherited from CIs, are not possessed by the two inputted generators. We have shown that, in addition of being chaotic, this generator can pass the NIST battery of tests, widely considered as a comprehensive and stringent battery of tests for cryptographic applications [13]. Then we have achieved to improve the speed of the former generator in [2], [4], by using ISAAC and XORshift instead of the two logistic maps. These generators can pass the batteries DieHARD [10] and TestU01 [14].

In these previous researches, the iteration function of CIs was always the vectorial Boolean negation. We propose now to enlarge the set of iteration functions such that the associated CI-based generator is both chaotic and random-like. The well-known NIST and DieHARD tests are finally used to evaluate the statistical behavior of this new family of generators.

The rest of this paper is organized as follows. In the next section, some basic definitions concerning CIs and our PRNG are recalled. In Section III is explained how it is possible to change the iteration function of the generator without losing the good properties of our PRNG. NIST and DieHARD batteries are passed in Section IV to all of these generators. The paper ends with a conclusion section where our contribution is summarized and intended future work is presented.

II. Review of Basics

This section is devoted to basic notations and terminologies in the fields of chaotic iterations and PRNGs.

A. Notations

$\llbracket 1;\mathsf{N}\rrbracket \to \{1,2,\ldots,\mathsf{N}\}$
$S^n \to$ the n^{th} term of a sequence $S=(S^1,S^2,\ldots)$
$v_i \to$ the i^{th} component of a vector $v=(v_1,v_2,\ldots,v_n)$
strategy $\to$ a sequence which elements belong in $\llbracket 1;\mathsf{N}\rrbracket$
$\mathbb{S} \to$ the set of all strategies
$\mathbb{N}^* \to$ the set of positive integers {1,2,3,...}
$\mathbb{B} \to \{0,1\}$

B. Chaotic iterations

Definition 1 Let $f:\mathbb{B}^{\mathsf{N}}\longrightarrow\mathbb{B}^{\mathsf{N}}$ be an "iteration" function and $S\in\mathbb{S}$. Then, the so-called *chaotic iterations* are defined by [12] $x^0\in\mathbb{B}^{\mathsf{N}}$ and

$$\forall n\in\mathbb{N}^*,\forall i\in\llbracket 1;\mathsf{N}\rrbracket, x_i^n=\begin{cases} x_i^{n-1} & \text{if } S^n\neq i\\ f(x^{n-1})_{S^n} & \text{if } S^n=i.\end{cases}$$

In other words, at the n^{th} iteration, only the S^n-th cell is "iterated".

C. Mapping matrix

Chaotic iterations introduced above can be described by using the mapping matrix defined bellow.

Definition 2 Let $f:\mathbb{B}^{\mathsf{N}}\longrightarrow\mathbb{B}^{\mathsf{N}}$ be an iteration function, then its associated *mapping matrix* f is the matrix of size $\mathsf{N}\times 2^{\mathsf{N}}$ whose element $\mathsf{f}_{p,q}$ is the integer having the following binary decomposition: $q_{\mathsf{N}},\ldots,q_{\mathsf{N}-p},f(q)_{\mathsf{N}-p+1},q_{\mathsf{N}-p+2},\ldots,q_1$, where q_i (resp. $f(q)_i$) is the $i-$th binary digit of q (resp. of $f(q)$).

The relation between f and chaotic iterations of f can be understood as follows. If the current state of the system is q and the strategy is p, then the next state (under the chaotic iterations of f) will be $\mathsf{f}_{p,q}$. Finally, the vector $\mathcal{F}(f)=(f(0),f(1),\ldots,f(2^{\mathsf{N}}-1))\in\llbracket 0;2^{\mathsf{N}}-1\rrbracket^{2^{\mathsf{N}}}$ is called *vector of images*. An example is shown for the vectorial Boolean negation $f_0(x_1,\ldots,x_{\mathsf{N}})=(\overline{x_1},\ldots,\overline{x_{\mathsf{N}}})$ in Table I.

TABLE I: The matrix f associated to f_0

p \ q	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$(f_0(q)_1, q_2, q_3, q_4)$	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
$(q_1, f_0(q)_2, q_3, q_4)$	4	5	6	7	0	1	2	3	12	13	14	15	8	9	10	11
$(q_1, q_2, f_0(q)_3, q_4)$	2	3	0	1	6	7	4	5	10	11	8	9	14	15	12	13
$(q_1, q_2, q_3, f_0(q)_4)$	1	0	3	2	5	4	7	6	9	8	11	10	13	12	15	14
$\mathcal{F}(f_0)$	(15,	14,	13,	12,	11,	10,	9,	8,	7,	6,	5,	4,	3,	2,	1,	0)

D. Chaotic iterations as PRNG

Algorithm 1 recalls the basic design procedure of our $CI_f(PRNG1, PRNG2)$ generator. The internal state is x, the output array is r, and $\mathsf{N} \in \mathbb{N}, \mathsf{N} \geqslant 2$. Parameters k and N are constants, PRNG1 picks its values into $\{0;1\}$, and PRNG2 takes a random integer into $[\![1;\mathsf{N}]\!]$. We have previously established that k must be greater than $3\mathsf{N}$ (see [1]). Finally, until now, $f = f_0$.

Input: the internal state x (N bits)
Output: a state r of N bits
$m \leftarrow PRNG1() + k$;
for $i = 0, \ldots, m$ **do**
  $S \leftarrow PRNG2()$;
  $x_S \leftarrow f(x)_S$;
end
$r \leftarrow x$;
return r;

Algorithm 1: An arbitrary round of $CI_f(PRNG1, PRNG2)$

This $CI_f(PRNG1, PRNG2)$ generator may utilize any reasonable PRNGs as inputs. For demonstration purposes, two XORshift are adopted here for both PRNG1 and PRNG2. Table II gives an illustrative example using these PRNGs, where $\mathsf{N} = k = 4$ and $\mathcal{F}(f)$ = (14, 14, 12, 12, 10, 10, 9, 9, 6, 6, 4, 4, 2, 2, 1, 0).

III. Description of the selection scheme

In this section is explained how the iteration function f_0 can be replaced without losing chaos and randomness.

A. Strong connectivity and chaos

Let $f : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}}$. Its *iteration graph* $\Gamma(f)$ is the directed graph defined as follows. The set of vertices is $\mathbb{B}^{\mathsf{N}}$, and $\forall x \in \mathbb{B}^{\mathsf{N}}, \forall i \in [\![1;\mathsf{N}]\!]$, $\Gamma(f)$ contains an arc labeled i from $x = (x_1, \ldots, x_{\mathsf{N}})$ to $(x_1, \ldots, x_{i-1}, f(x)_i, x_{i+1}, \ldots, x_{\mathsf{N}})$. We have proven in [7] that:

Theorem 1 *The $CI_f(PRNG1, PRNG2)$ generator is chaotic according to Devaney if and only if the graph $\Gamma(f)$ is strongly connected.*

Theorem 1 only focus on the topological chaos property. However, it is possible to find chaotic sequences with bad statistical properties, in particular when the iteration function is unbalanced.

B. Obtaining Balanced Maps

We now explain how to find balanced iterate functions.

Theorem 2 *Let $j \in [\![1; 2^{\mathsf{N}}]\!]$ and $F = \mathcal{F}(f_0)$ be the (balanced) vectorial Boolean negation: $F_j = 2^{\mathsf{N}} - j$.*

If $F' = \mathcal{F}(f)$, a vector of images of a balanced *iterate function f, is such that its $j-th$ component differs from F_j by only its $i-th$ bit (starting from the right), then $F'_{2^{\mathsf{N}}-F'_j} = 2^{\mathsf{N}} - j$.*

Proof: As F'_j only differs from F_j by its $i-th$ bit, we have: $F'_j = F_j - F_j \& 2^{i-1} + (j-1)\& 2^{i-1}$. Therefore, the value $\mathsf{f}'_{i,j}$ of the mapping matrix of F' can be computed as follows:

$$\begin{aligned}
&\mathsf{f}'_{i,j} = j_{\mathsf{N}} j_{\mathsf{N}-1} \ldots f(j)_i \ldots j_1 \\
&= (j-1) - (j-1)\&2^{i-1} + F'_j \& 2^{i-1} \\
&= (j-1) - (j-1)\&2^{i-1} + F_j \& 2^{i-1} - F_j \& 2^{i-1} + \\
&\qquad\qquad (j-1)\&2^{i-1} \\
&= (j-1)
\end{aligned} \tag{1}$$

The values in f are uniformly distributed. However, in the new matrix N, there are twice the value $j-1$ and no $\mathsf{f}_{i,j}$ in the i-th row: the uniform distribution is lost. To restore the balance, one of the two $j-1$ values must be found and replaced by $\mathsf{f}_{i,j}$. Let k be a variable such that $\mathsf{f}_{i,k} = j-1$ and $\mathsf{f}'_{i,k} = \mathsf{f}_{i,j}$.

As the $i-$th bits in $f_{i,k}$ and $f'_{i,k}$ are equal, we have:

$$\mathsf{f}'_{i,k} \& (2^{\mathsf{N}} - 1 - 2^{i-1}) = \mathsf{f}_{i,j} \& (2^{\mathsf{N}} - 1 - 2^{i-1}). \tag{2}$$

We can thus transform the equation $\mathsf{f}_{i,k} = j-1$ as follows:

$$\begin{aligned}
\mathsf{f}_{i,k} &= j-1 \\
(k-1) - (k-1)\&2^{i-1} + F_k \& 2^{i-1} &= j-1 \\
(k-1)\&(2^{\mathsf{N}} - 1 - 2^{i-1}) + F_k \& 2^{i-1} &= j-1.
\end{aligned} \tag{3}$$

Moreover, from $F_k \& 2^i = 2^{\mathsf{N}} - 1 - k$, we obtain:

$$\begin{aligned}
F_k \& 2^{i-1} &= (j-1)\&2^{i-1} \\
(2^{\mathsf{N}} - k)\&2^i &= (j-1)\&2^{i-1} \\
(k-1)\&2^{i-1} &= (k-1)\&2^{i-1} - (j-1)\&2^{i-1}.
\end{aligned} \tag{4}$$

According to Equations (3) and (4), we have:

$$\begin{aligned}
&k-1 = (j-1) + (k-1)\&2^{i-1} - F_k \& 2^{i-1}, \\
&\text{where } F_k = 2^{\mathsf{N}} - k \\
&= (j-1) + (k-1)\&2^{i-1} - 2^{i-1} + (k-1)\&2^{i-1} \\
&= (j-1) - 2^{i-1} + ((k-1)\&2^{i-1}) * 2. \\
&\text{But, due to Equation}(5), \text{ we have:} \\
&= (j-1) - 2^{i-1} + ((k-1)\&2^{i-1} - (j-1)\&2^{i-1}) * 2 \\
&= (j-1) + 2^{i-1} - ((j-1)\&2^{i-1}) * 2 \\
&= (j-1) + (j-1)\&2^{i-1} + (2^{\mathsf{N}} - j)\&2^{i-1}, \\
&\text{where } F_j = 2^{\mathsf{N}} - 1 - j \\
&= (j-1) + (j-1)\&2^{i-1} + F_j \& 2^{i-1} \\
&= \mathsf{f}_{i,j}.
\end{aligned} \tag{5}$$

As

$$\mathsf{f}'_{i,k} = (k-1) - (k-1)\&2^{i-1} + F'_k \& 2^{i-1} \tag{6}$$

and according to Equation (5), we thus have:

$$F'_k \& 2^{i-1} = (k-1)\&2^{i-1}. \tag{7}$$

m :	4					5						4				
S	2	4	2	3		4	1	1	4	4		3	2	3	3	
	1	1	1	1		1	**1**	**0**	1	1		1	1	1	1	
f(x)	**0**	1	**1**	0		0	0	0	0	0		0	**0**	1	1	
	1	1	1	**1**		0	0	0	0	0		**0**	1	**1**	**0**	
	0	**0**	0	0		**1**	1	0	**1**	**1**		1	0	0	0	
x^0					x^4						x^9					x^{13}
4	0	0	4	6	6	7	15	7	7	7	7	5	1	3	1	1
0					0		$\xrightarrow{1}1$	$\xrightarrow{1}0$			0					0
1	$\xrightarrow{2}0$		$\xrightarrow{2}1$		1						1		$\xrightarrow{2}0$			0
0				$\xrightarrow{3}1$	1						1	$\xrightarrow{3}0$		$\xrightarrow{3}1$	$\xrightarrow{3}0$	0
0		$\xrightarrow{4}0$			0	$\xrightarrow{4}1$			$\xrightarrow{4}1$	$\xrightarrow{4}1$	1					1

Binary Output: $x_1^0x_2^0x_3^0x_4^0x_5^0x_1^4x_2^4x_3^4x_4^4x_5^4x_1^9x_2^9x_3^9x_4^9x_5^9x_1^{13}x_2^{13}... = 0100011001110001...$
Integer Output: $x^0, x^0, x^4, x^6, x^8... = 6, 7, 1...$

TABLE II: Application example

Now, from Equation (2), we can set that:

$$\begin{aligned}
&\mathsf{f}'_{i,k}\&(2^{\mathsf{N}}-1-2^{i-1}) = \mathsf{f}_{i,j}\&(2^{\mathsf{N}}-1-2^{i-1})\\
&((k-1)-(k-1)\&2^{i-1}+F'_k\&2^{i-1})\&(2^{\mathsf{N}}-1-2^{i-1}) =\\
&\quad((k-1)-(k-1)\&2^{i-1}+F_k\&2^{i-1})\&(2^{\mathsf{N}}-1-2^{i-1})\\
&F'_k\&(2^{\mathsf{N}}-1-2^i) = F_k\&(2^{\mathsf{N}}-1-2^i).
\end{aligned} \tag{8}$$

By using both Equations (7) and (8), we obtain:

$$\begin{aligned}
F'_k &= (k-1)\&2^{i-1}+F_k\&(2^{\mathsf{N}}-1-2^{i-1})\\
&= (k-1)\&2^{i-1}+F_k-F_k\&2^{i-1}\\
&= \mathsf{f}_{i,j}\&2^{i-1}+(2^N-1-\mathsf{f}_{i,j})-(2^N-1-\mathsf{f}_{i,j})\&2^{i-1}\\
&= (2^N-1)-[\mathsf{f}_{i,j}+2^{i-1}-2\times(\mathsf{f}_{i,j}\&2^{i-1})]\\
&= (2^N-1)-((j-1)-(j-1)\&2^{i-1}+F_j\&2^{i-1}\\
&\quad+2^{i-1}-2\times[((j-1)-(j-1)\&2^{i-1}+F_j\&2^{i-1})\\
&\qquad\qquad\&2^{i-1}\\
&= (2^N-1)-((j-1)-(j-1)\&2^{i-1}-F_j\&2^{i-1}\\
&\qquad\qquad+2^{i-1})\\
&= (2^N-1)-((j-1)+2^{i-1}\&(2^N-j)-F_j\&2^{i-1})\\
&= (2^N-1)-((j-1)+F_j\&2^{i-1}-F_j\&2^{i-1})\\
&= (2^N-1)-(j-1)\\
&= 2^N-j.
\end{aligned} \tag{9}$$

Finally, from Equation (5), we can conclude that:

$$\begin{aligned}
k &= \mathsf{f}_{i,j}+1\\
&= (j-1)-(j-1)\&2^{i-1}+F_j\&2^{i-1}\\
&= 2^N-(2^N-j)-(j-1)\&2^{i-1}+F_j\&2^{i-1}\\
&= 2^N-(F_j-F_j\&2^i+(j-1)\&2^{i-1})\\
&= 2^N-F'_j.
\end{aligned} \tag{10}$$

■

With such equations (namely, Eq. (9) and (10)), the balance of the new table can be obtained by computing the mapping values. In other words, there is a bijection from the set A of the inputs x into the set B of $F'(x)$ values.

Let us give an example. In Table I is given the mapping matrix for the vectorial Boolean negation, with $\mathsf{N} = 4$. Obviously, the values in f are uniformly distributed: each integer from 0 to 15 occurs once per row. Now, if we desire to set F'_1 as 14, then $\mathsf{f}'_{4,1} = 0$: there will be two 0 and no 1 in the fourth row of f'. Due to the previous study, we know that F'_2 must be set to 15 too, which leads to $\mathsf{f}'_{4,2} = 1$: the balance is recovered.

To sum up, we can determine whether the modification of a bit in the vector of images of the negation function preserves the balance of the outputs or not, by using the following rule (necessary condition):

- if $F'_j = C$,
- then $C = F_j - F_j\&2^{i-1} + (j-1)\&2^{i-1}$,
- and also $F_{2^N-C} = 2^N - j$.

This rule, we name it "Balance Iteration Mapping Rule", can be used as a criterion to find iterate functions leading to good CI PRNGS, as it is depicted in Algorithm 2. Let us finally remark that, with such a process, it is possible to find new iteration functions by changing more than 1 couple of values in the vectorial Boolean negation F. Indeed it is obvious that 2, 3, 4, and even 8 couples of values can be changed using the Balance Iteration Mapping Rule. For instance, Table III contains 8 vectors of images obtained by using Algorithm 2 one or more times. All of these functions satisfy the hypothesis of Theorem 1 too, and thus their dynamical systems behave chaotically.

Input: a vector of images F
Output: a vector of images r or 0
for $i = 0, \ldots, 2^{\mathsf{N}} - 1$ **do**
  for $j = 0, \ldots, N$ **do**
    if $F(i+1) \neq F(i+1) - F(i+1)\&2^j + i\&2^j$ **then**
      if $F(i+1) \neq 2^N - 1 - i$ **then**
        return 0;
      end
    end
  end
  if $F(2^N - F(i)) \neq 2^N - i$ **then**
    return 0;
  end
end
return F;

Algorithm 2: The Balance Iteration Mapping Rule.

IV. Statistical analysis

A good random number generator must be indistinguishable from a random sequence through any statistical test. As an illustration of the theory presented in this paper, we have used various batteries of tests in order to evaluate the quality of our proposed pseudo random number generator, when iterating

TABLE IV: Results through NIST SP 800-22 and DieHARD batteries of tests ($\mathbb{P}_T$ values)

Method	F'_1	F'_2	F'_3	F'_4	F'_5	F'_6	F'_7	F'_8
Frequency (Monobit) Test	0.102526	0.017912	0.171867	0.779188	0.971699	0.275709	0.137282	0.699313
Frequency Test within a Block	0.085587	0.657933	0.779188	0.897763	0.851383	0.383827	0.262249	0.122325
Cumulative Sums (Cusum) Test*	0.264576	0.185074	0.228927	0.736333	0.462694	0.169816	0.391715	0.729111
Runs Test	0.739918	0.334538	0.798139	0.834308	0.153763	0.719747	0.534146	0.262249
Test for the Longest Run of Ones in a Block	0.678686	0.474986	0.637119	0.037566	0.366918	0.739918	0.236810	0.759756
Binary Matrix Rank Test	0.816537	0.534146	0.249284	0.883171	0.739918	0.037566	0.798139	0.867692
Discrete Fourier Transform (Spectral) Test	0.798139	0.474986	0.014550	0.366918	0.595549	0.115387	0.798139	0.153763
Non-overlapping Template Matching Test*	0.489304	0.507177	0.477005	0.557597	0.452278	0.505673	0.541034	0.497140
Overlapping Template Matching Test	0.514124	0.171867	0.162606	0.816537	0.319084	0.678686	0.534146	0.798139
Maurers Universal Statistical Test	0.249284	0.171867	0.096578	0.419021	0.171867	0.798139	0.115387	0.275709
Approximate Entropy Test	0.236810	0.514124	0.262249	0.816537	0.474986	0.080519	0.000001	0.779188
Random Excursions Test*	0.353142	0.403219	0.229832	0.481025	0.317506	0.602978	0.362746	0.416274
Random Excursions Variant Test*	0.412987	0.369181	0.313171	0.513679	0.274813	0.391166	0.454157	0.341012
Serial Test* (m=10)	0.304324	0.102735	0.270033	0.384058	0.456684	0.125973	0.404429	0.253197
Linear Complexity Test	0.759756	0.153763	0.883171	0.171867	0.366918	0.319084	0.678686	0.075719
Success	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15
Diehard Test	pass	pass	pass	pass	pass	pass	pass	pass

Name	Map
F	[15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0]
$F'1$	[14,15,13,12,11,10,9,8,7,6,5,4,3,2,1,0]
$F'2$	[14,15,13,12,9,10,11,8,7,6,5,4,3,2,1,0]
$F'3$	[14,15,9,4,11,8,13,10,7,6,5,12,3,2,1,0]
$F'4$	[14,15,9,12,3,8,13,10,7,6,5,4,11,2,1,0]
$F'5$	[14,15,9,4,11,8,13,10,7,6,5,12,3,2,0,1]
$F'6$	[14,15,9,4,11,8,13,10,3,6,5,12,7,2,0,1]
$F'7$	[14,15,9,4,3,8,13,10,5,2,7,12,11,6,1,0]
$F'8$	[14,15,5,8,9,2,11,12,3,4,13,6,7,10,0,1]

TABLE III: New vectors of images

functions of Table III. These batteries are the well-known and stringent DIEHARD [10] and NIST [13] statistical test suites.

We can conclude from Table IV that all of the generators based on the new iterate functions have successfully passed both the NIST and DieHARD batteries of tests. These results show the good statistical properties of the proposed PRNGs, and thus the interest of the theoretical approach presented in this paper.

V. Conclusion and future work

In previous researches, we have presented a pseudo-random number generator based on chaotic iterations. It depends on an iteration function, formerly fixed to the negation function. We have previously established a characterization of functions leading to a chaotic behavior for the associated generator. However, this characterization allows unbalanced functions, whose generator cannot pass statistical tests. We have proposed in this paper an algorithm that can find iteration functions leading to a chaotic generator statistically irreproachable. This algorithm has been used to find 8 functions such that their generators are both chaotic and able to pass the NIST and DIEHARD statistical batteries of tests.

In future work, we will continue to explore conditions that improve the randomness of the associated CI PRNGs. New statistical tests will be used to compare these PRNGs to existing ones, and a cryptanalysis of our generator will be proposed. Finally, new applications in computer science will be proposed, especially in the Internet security field.

References

[1] Jacques Bahi, Christophe Guyeux, and Qianxue Wang. A novel pseudo-random generator based on discrete chaotic iterations. In *INTERNET'09, 1-st Int. Conf. on Evolving Internet*, pages 71–76, Cannes, France, August 2009.

[2] Jacques Bahi, Christophe Guyeux, and Qianxue Wang. A pseudo random numbers generator based on chaotic iterations. application to watermarking. In *WISM 2010, Int. Conf. on Web Information Systems and Mining*, volume 6318 of *LNCS*, pages 202–211, Sanya, China, October 2010.

[3] Jacques M. Bahi and Christophe Guyeux. Hash functions using chaotic iterations. *Journal of Algorithms & Computational Technology*, 4(2):167–181, 2010.

[4] Jacques M. Bahi, Christophe Guyeux, and Qianxue Wang. Improving random number generators by chaotic iterations. application in data hiding. In *ICCASM 2010, Int. Conf. on Computer Application and System Modeling*, pages V13–643–V13–647, Taiyuan, China, October 2010.

[5] S. Behnia, A. Akhavan, A. Akhshani, and A. Samsudin. A novel dynamic model of pseudo random number generator. *Journal of Computational and Applied Mathematics*, 235(12):3455–3463, 2011.

[6] Robert L. Devaney. *An Introduction to Chaotic Dynamical Systems*. Addison-Wesley, Redwood City, CA, 2nd edition, 1989.

[7] Christophe Guyeux. *Le désordre des itérations chaotiques et leur utilité en sécurité informatique*. PhD thesis, Université de Franche-Comté, 2010.

[8] Yue Hu, Xiaofeng Liao, Kwok wo Wong, and Qing Zhou. A true random number generator based on mouse movement and chaotic cryptography. *Chaos, Solitons & Fractals*, 40(5):2286–2293, 2009.

[9] N. Liu. Pseudo-randomness and complexity of binary sequences generated by the chaotic system. *Communications in Nonlinear Science and Numerical Simulation*, 16(2):761–768, 2011.

[10] George Marsaglia. Diehard: a battery of tests of randomness. *http://www.stat.fsu.edu/pub/diehard/*, 1996.

[11] L. De Micco, C.M. Gonzaez, H.A. Larrondo, M.T. Martin, A. Plastino, and O.A. Rosso. Randomizing nonlinear maps via symbolic dynamics. *Physica A: Statistical Mechanics and its Applications*, 387(14):3373–3383, 2008.

[12] F. Robert. *Discrete Iterations: A Metric Study*, volume 6. Springer Series in Computational Mathematics, 1986.

[13] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications. *NIST Special Publication 800-22*, 2010.

[14] Richard Simard and Université De Montréal. Testu01: A software library in ansi c for empirical testing of random number generators. software users guide. *ACM Transactions on Mathematical Software*, 2002.